APLIKASI EMAIL MENGGUNAKAN J2ME

Wilis Kaswidjanti, Novrido Charibaldi, Fitri Hidayati
Jurusan Teknik Informatika UPN "Veteran" Yogyakarta
Jl. Babarsari 2 Tambakbayan 55281 Telp (0274) 485323
Email :wilisk@yahoo.com, novrido@gmail.com

Abstrak
Pada saat ini perkembangan telepon seluler yang mendukung koneksi GPRS (General Packet Radio Service) dan teknologi java semakin berkembang pesat. Aplikasi java telah banyak dikembangkan pada telepon seluler sebagai penunjang aktifitas sehari-hari. Telepon seluler merupakan pilihan aplikasi komunikasi dengan gerak luas, tetapi memerlukan biaya yang mahal. Pada telepon seluler alternatif untuk melakukan komunikasi lebih murah adalah SMS (Short Message Service). Tetapi SMS mempunyai keterbatasan dalam jumlah karakter isi untuk pengiriman dan untuk mengirimkan SMS antar operator telepon apalagi luar negri memerlukan biaya yang mahal. Untuk mengatasi masalah tersebut maka pada tugas akhir ini dibangun aplikasi pengiriman maupun membaca email melalui telepon seluler dengan memanfaatkan SMTP dan POP3 server.
Metode yang digunakan dalam perancangan dan pembuatan perangkat lunak ini adalah metode RUP (Rational Unified Process).Adapun perangkat lunak yang digunakan adalah netbeans IDE 6.1. Aplikasi telepon seluler menggunakan J2ME. Bahasa pemodelan sistem yang digunakan adalah UML (Unifield Modelling Language).
Penelitian ini bertujuan untuk menghasilkan aplikasi yang memudahkan proses dan meminimalkan biaya pengiriman dalam berkomunikasi menggunakan telepon seluler sehingga email tersebut dapat dibaca dan dikirimkan kapan saja dengan memanfaatkan teknologi GPRS antara client dan server mail.

Katakunci : Aplikasi, Email, J2ME

1. PENDAHULUAN

Perkembangan teknologi di bidang komunikasi serta komputer yang pesat makin memudahkan masyarakat dalam berkomunikasi. Dengan adanya dunia internet yang menghadirkan berbagai pilihan cara untuk berkomunikasi secara langsung maupun tidak langsung seperti *email*, *chatting*, ftp, telnet dan lain sebagainya. Dapat dilakukan dengan biaya murah, tetapi untuk melakukan komunikasi tersebut pengguna harus menggunakan komputer yang ruang geraknya terbatas.

Layanan yang dapat digunakan pada *handphone* untuk melakukan komunikasi dengan biaya yang relatif murah adalah SMS (*Short Message Service)*. Tetapi SMS mempunyai keterbatasan dalam jumlah karakter pesan.

Sejalan dengan keperluannya maka perkembangan teknologi pada *handphone* terus bertambah, salah satu contohnya adalah penerapan teknologi java pada *handphone*. Hal ini memungkinkan penerapan aplikasi yang ditulis dengan bahasa pemrograman java pada *handphone*. Dengan adanya fungsi tersebut maka berbagai macam aplikasi dapat dibuat dan dijalankan pada *handphone*. Aplikasi pengiriman maupun membaca *email* melalui *handphone* dapat dibuat untuk mengatasi masalah diatas.

2. TINJAUAN PUSTAKA

Aplikasi

Aplikasi merupakan program yang khusus dibuat untuk melakukan suatu pekerjaan atau proses tertentu. Biasanya program dibuat oleh seorang *programmer* komputer yang disesuaikan dengan permintaan atau kebutuhan seseorang, lembaga atau perusahaan (Kadir, 2002).

Email

Email atau *electronic mail* merupakan layanan berupa pengiriman pesan teks yang datanya diubah ke bentuk data elektronik dan dikirimkan melalui jaringan komputer (Prasetyo,2004).

SMTP (*Simple Mail Transfer Protocol*) merupakan protokol yang digunakan untuk mengirim *email*, dan bekerja pada port 25 (Prasetyo,2004). Pengiriman *email* dilakukan menggunakan aplikasi MTA, misalnya *kerio, mdaemon, sendmail, qmail*, dan sebagainya. Protokol ini merupakan protokol yang sekarang banyak dipakai oleh *mail server* (MTA) di internet.

POP3 (*Post Office Protocol Version 3*) adalah protokol yang digunakan untuk mengambil surat elektronik (*email*) dari *server email* (Prasetyo,2004). Protokol ini sangat erat hubungannya dengan protokol SMTP (*Simple Mail Transfer Protocol*) dimana protokol SMTP berguna untuk mengirim surat elektronik dari komputer pengirim ke komputer *server* dari penerima. Kemudian penerima mengambil surat elektronik yang dikirim dari *server* dengan menggunakan protokol ini.

GPRS

GPRS (*General Packet Radio Services*) adalah suatu teknologi yang memungkinkan pengiriman dan penerimaan data lebih cepat jika dibandingkan dengan penggunaan teknologi CSD (*Circuit Switch Data*). (http://id.wikipedia.org/wiki/GPRS)

RUP

Rational Unified Process (RUP) merupakan(Kroll dan Kruchten, 2003):

a. Pendekatan pengembangan perangkat lunak yang bersifat iteratif, bertitik berat pada arsitektur dan *use case driven*.
b. Proses pembangunan perangkat lunak yang terdefinisi dan terstruktur dengan baik. RUP dengan jelas mendefinisikan siapa (*who*) yang bertanggung jawab terhadap sesuatu (*what*), bagaimana (*how*) pekerjaan dilakukan dan kapan (*when*) mengerjakannya. RUP juga menyediakan struktur yang terdefinisi dengan baik selama siklus hidup proyek dengan menentukan tahapan-tahapan penting (*milestone*) serta titik-titik utama yang menjadi fokus pengambilan keputusan.
c. Suatu perangkat lunak yang menyediakan lingkup kerja atau kerangka proses yang fleksibel bagi pembangunan perangkat lunak.

Java

Java adalah bahasa pemrograman serbaguna. Dikembangkan oleh *Sun Microsystem* pada Agustus 1991, dengan nama semula Oak. Konon Oak adalah pohon semacam jati yang terlihat dari jendela tempat pembuatnya, James Gosling, bekerja. Ada yang mengatakan Oak adalah *Object Aplication Kernel*, tetapi ada yang menyatakan hal itu muncul setelah nama Oak diberikan. Pada Januari 1995, karena nama Oak dianggap kurang komersial, maka diganti menjadi Java (Kadir, 2004). J2ME (*Java 2 Micro Edition*) adalah lingkungan pengembangan yang didesain untuk meletakkan perangkat lunak Java pada barang elektronik berserta perangkat pendukungnya (Shalahuddin dan Rosa, 2006).

UML

UML (Unified Modelling Language) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek (Munawar, 2005). UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modeling Technique* (OMT) dan *Object Oriented Software Enginering* (OOSE) (Munawar, 2005).

3. ANALISIS DAN PERANCANGAN

Requirements bertujuan untuk mendeskripsikan apa yang harus dilakukan oleh sistem. Terdapat dua jenis *requirements* yang harus dilakukan untuk menghasilkan sebuah sistem yang berkualitas, yaitu kebutuhan fungsional dan kebutuhan non-fungsional. *Requirements* dilakukan pertama kali pada fase *inception*.

Kebutuhan Fungsional

Beberapa kebutuhan fungsional untuk pengguna adalah:

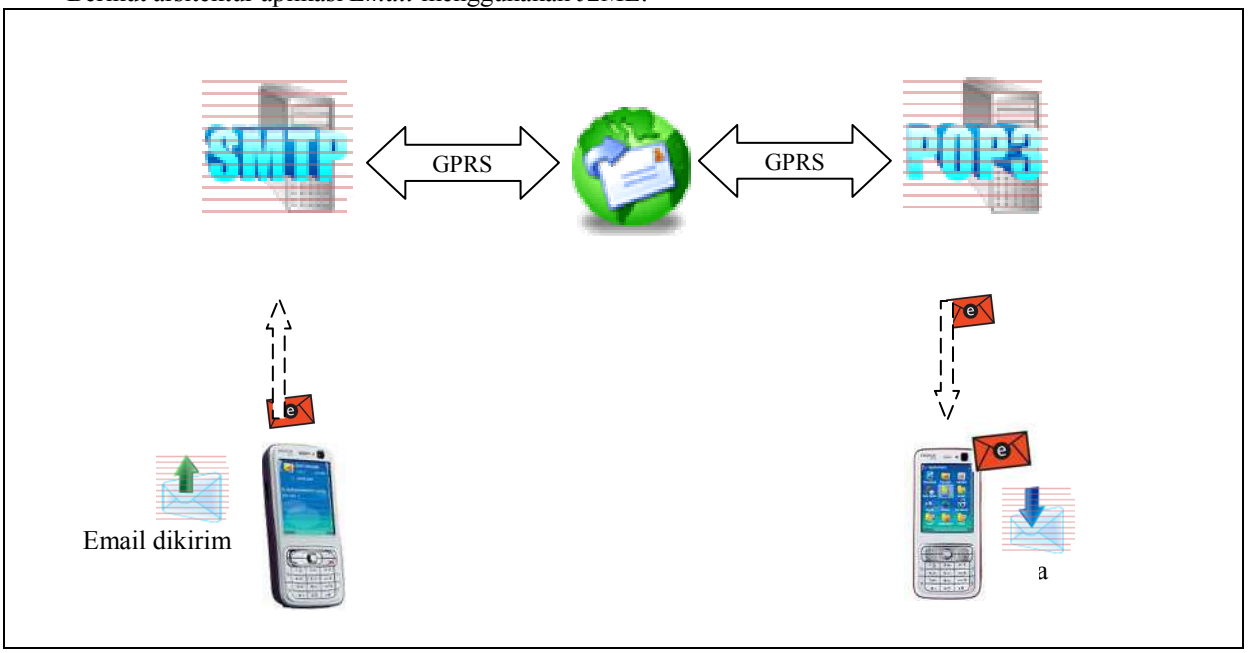
1. Terdapat pengaturan konfigurasi POP3 dan SMTP.
2. Aplikasi dapat mengirimkan dan menerima email secara langsung melalui *Handphone*.

Kebutuhan Non-Fungsional

Beberapa kebutuhan non-fungsional dalam aplikasi yang dibangun adalah *user* dapat mengakses aplikasi melalui koneksi via GPRS dan terdapat antarmuka yang mudah dipelajari dan mudah digunakan.

Arsitektur Sistem

Berikut arsitektur aplikasi *Email* menggunakan J2ME:

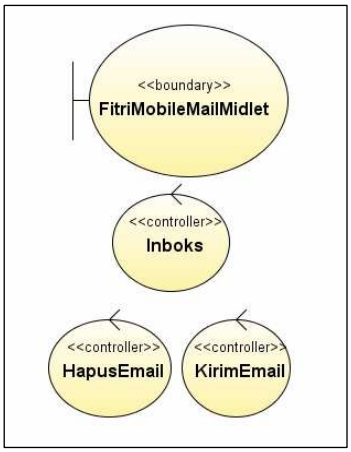


Gambar 1. Arsitektur Sistem Aplikasi *Email* menggunakan J2ME

Penjelasan pada gambar diatas adalah aplikasi ini *User* dapat mengirimkan *email* dengan melakukan koneksi SMTP terlebih dahulu, kemudian *email* tersebut dikirimkan dari SMTP *server* menuju *Mail server* dengan melakukan koneksi GPRS. Untuk dapat menerima *email* tersebut *email* yang berada pada *email server* dikirimkan menuju POP3 *server* mengunakan koneksi GPRS, kemudian *email* yang berada pada POP3 *server* akan dikirimkan pada *User* penerima melalui koneksi POP3.

High Level Class

Dari permasalahan yang ada, terdapat empat *class* yang akan dikembangkan dalam tahapan selanjutnya, yang terdiri dari sebuah *Boundary class* dan tiga *Control cla*ss.



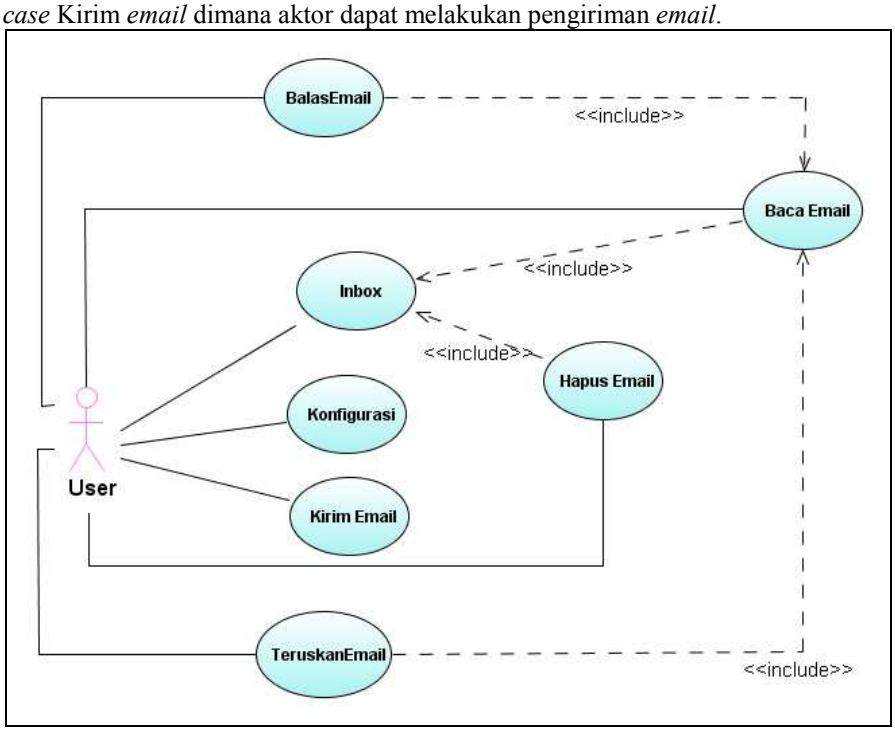
Gambar 2. Diagram *High Level Class* Aplikasi *Email* menggunakan J2ME.

Analysis

Analysis dilakukan pertama kali pada fase *inception*.

Diagram *Use Case*

Proses pengembangan perangkat lunak pada RUP/UP berlangsung melalui serangkaian aktivitas yang diturunkan dari *use case*, sehingga diagram UML yang harus dibuat pertama kali adalah diagram *use case*. Berdasarkan hasil analisis terhadap *requirements*, maka didapatkan satu aktor yaitu *user* yang berinteraksi dengan 7 *use case*, yaitu *use case Inbox* dimana aktor dapat menerima *email*, *use case* Baca *email* dimana aktor dapat membaca *email* yang masuk, *use case* Balas *email* dimana aktor dapat membalas *email* yang masuk, *use case* Teruskan *email* dimana aktor dapat mem*forward email* yang masuk, *use case* Hapus *email* dimana aktor dapat menghapus *email* yang diinginkan, *use case* Konfigurasi dimana aktor dapat mengatur konfigurasi yang sesuai dan *use case* Kirim *email* dimana aktor dapat melakukan pengiriman *email*.



Gambar 3. Diagram *Use Case* dalam Aplikasi *Email* menggunakan J2ME

Reverse Engineering

Reverse Engineering merupakan kemampuan Netbeans 6.1 yang dapat membuat diagram UML secara otomatis melalui *coding* java yang sudah jadi (Netbeans 6.1 *Help*).

Dalam proyek pengembangan sistem apapun, fokus utama dalam analisis dan perancangan adalah model (Munawar, 2005). Hal ini berlaku umum tidak hanya untuk perangkat lunak. Dengan model bisa merepresentasikan sesuatu karena:

1. Model mudah dan cepat untuk dibuat.
2. Model bisa digunakan sebagai simulasi untuk mempelajari lebih detil tentang sesuatu.
3. Model bisa dikembangkan sejalan dengan pemahaman tentang sesuatu.
4. Dapat memberikan penjelasan lebih rinci tentang sesuatu dengan model.
5. Model bisa mewakili sesuatu yang nyata maupun tidak nyata.

Rancangan pemodelan pada penelitian ini menggunakan diagram UML (*Unified Modelling Language*). Diagram-diagram ini meliputi diagram *class*, diagram *sequence*, diagram *activity* yang dihasilkan dari proses *reverse engineering* pada Netbeans 6.1.

Diagram *Class*

Diagram *class* pada Aplikasi *Email* menggunakan J2ME dibuat menggunakan *Reverse Engineering* yang terdapat pada Netbeans 6.1. Diagram *class* ini menunjukan hubungan antar beberapa *class*. Terdapat empat *class* dalam aplikasi ini, yaitu *class FitriMobileMail* sebagai *class* utama, *class Inboks* yang berfungsi untuk menangani proses *download email*, *class HapusEmail* yang berfungsi untuk menangani proses hapus *email* dan *class KirimEmail* yang berfungsi untuk menangani kirim *email*. Gambar 4 merupakan gambar *class* diagram.

Diagram *Dependencies*

Diagram *Dependencies* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain (Munawar, 2005). Dari diagram *Control class* diatas akan dibuat diagram *Dependencies* yang nantinya diagram tersebut dapat di *reverse* kedalam diagram *Sequence*.

Diagram *Sequence*

Diagram *sequenc*e menggambarkan perilaku pada sebuah sekenario. Diagram ini menunjukkan sejumlah contoh objek dan pesan yang diletakkan diantara objek-objek ini di dalam *use case*. Terdapat tujuh diagram *sequence* dalam aplikasi ini :

a. Diagram *Sequence* Konfigurasi memperlihatkan event-event yang terjadi pada proses pengaturan konfigurasi dari pertama *user* membuka aplikasi sampai proses bagaimana *user* mengisi *textField* pada konfigurasi.
b. Diagram *Sequence* Inboks memperlihatkan event-event yang terjadi pada saat *user* melakukan proses *download email*.
c. Diagram *sequence* Baca *Email* memperlihatkan event-event yang terjadi pada saat *user* melakukan proses baca *email*.
d. Diagram *sequence* Balas *Email* memperlihatkan event-event yang terjadi pada saat *user* melakukan proses membalas *email*.
e. Diagram *sequence* Teruskan *Email* memperlihatkan event-event yang terjadi pada saat *user* melakukan proses *forward email*.
f. Diagram *sequence* Hapus *Email* memperlihatkan event-event yang terjadi pada saat *user* melakukan proses hapus *email*.
g. Diagram *sequence* Kirim *Email* memperlihatkan event-event yang terjadi pada saat user mengisi *textField* pada formKirim*Email* hingga user melakukan proses kirim *email*.

Diagram *Activity*

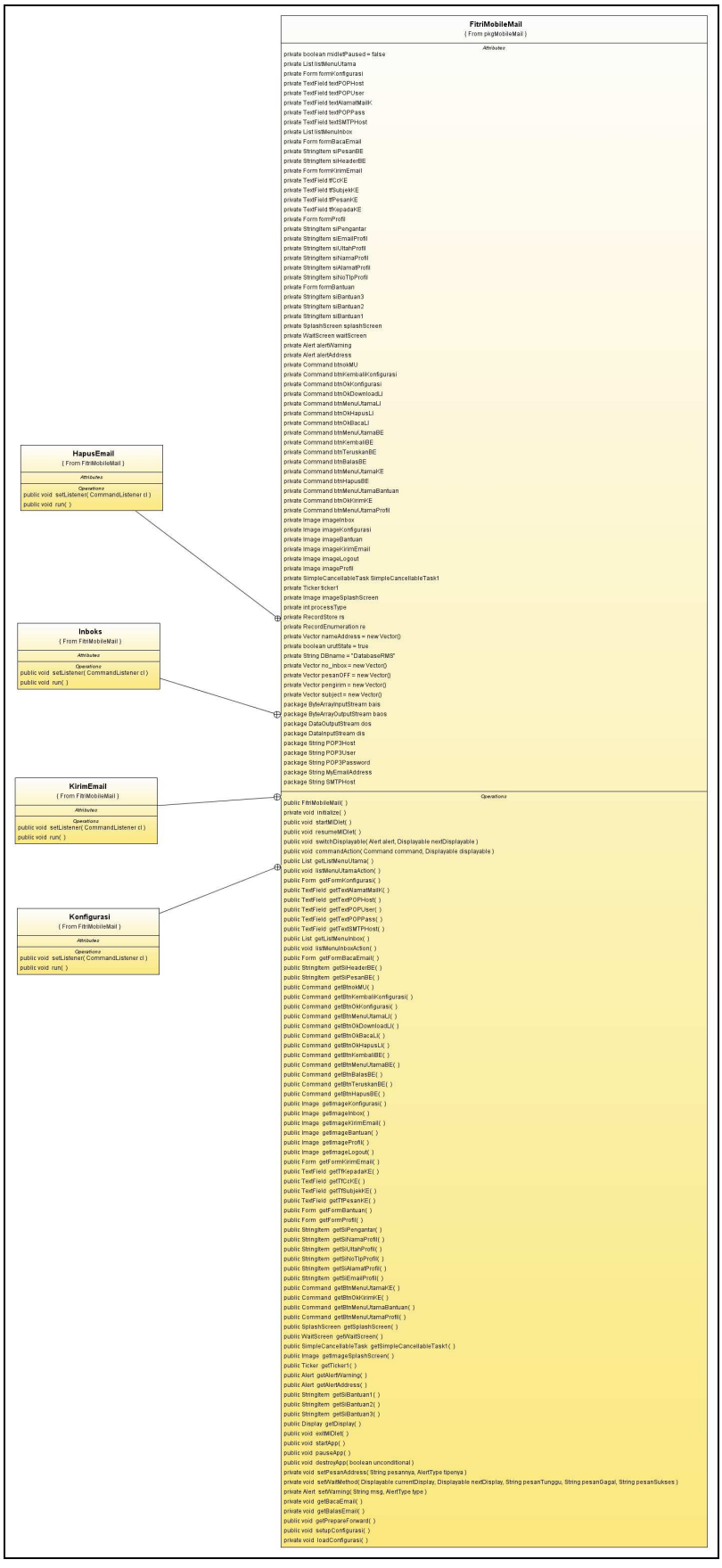
Diagram *activity* digunakan untuk memodelkan aspek dinamis dari aplikasi. Digram ini menggambarkan berbagai aliran aktivitas yang terjadi dalam aplikasi dari awal aktivitas sampai berakhirnya aktivitas yang terjadi pada masing-masing proses. *Activity* diagram mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity* diagram bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa (Munawar, 2005).

Diagram *Activity* Konfigurasi

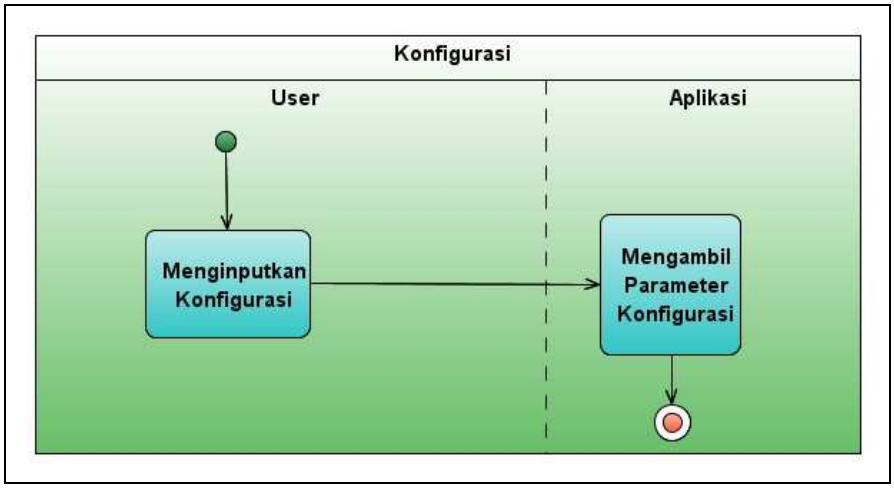
Diagram *activity* Konfigurasi menunjukkan serangkaian proses aktivitas *user* melakukan proses pengaturan konfigurasi. Dalam aktivitasnya terdapat dua *invocation* yang menunjukkan setiap proses ke proses berikutnya. Diagram ini memperlihatkan proses pengaturan konfigurasi yang dilakukan oleh *user*, kemudian aplikasi akan mengambil parameter dari konfigurasi tersebut (gambar 5).

Diagram *Activity* Inbox

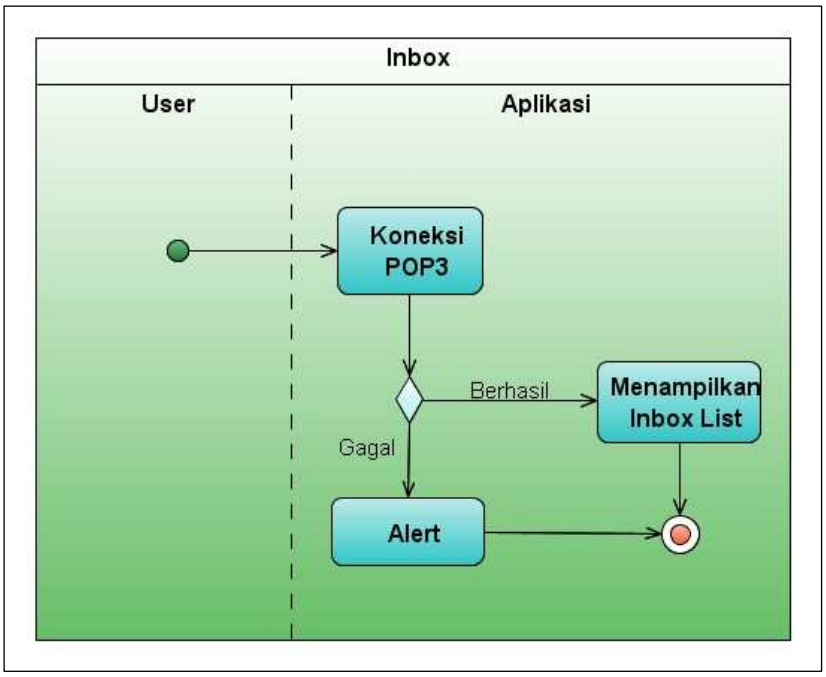
Diagram *activity* Inbox menunjukkan serangkaian proses aktivitas *user* melakukan proses pengambilan *email*. Dalam aktivitasnya terdapat tiga *invocation* yang menunjukkan setiap proses ke proses berikutnya dan satu *decision* untuk proses berhasil tidaknya dalam melakukan koneksi terhadap POP3. dimana jika koneksi berhasil maka akan menampilkan *Inbox list* yang berisi *email – email* yang telah berhasil di *download*, tetapi Jika koneksi gagal maka akan ditampilkan peringatan. Dapat dilihat pada gambar 6.



Gambar 4. Diagram *Class* Aplikasi Aplikasi *Email* Menggunakan J2ME.



Gambar 5. Diagram *Activity* Konfigurasi Aplikasi *Email* menggunakan J2ME.



Gambar 6. Diagram *Activity Inbox* Aplikasi *Email* menggunakan J2ME.

Diagram *Activity* BacaEmail

Diagram *activity* Baca *Email* menunjukkan serangkaian proses aktivitas *user* melakukan proses baca *email*. Dalam aktivitasnya terdapat empat *invocation* yang menunjukkan setiap proses ke proses berikutnya dan satu *decision* untuk proses berhasil tidaknya dalam melakukan koneksi terhadap POP3. Diagram ini memperlihatkan *user* melakukan koneksi POP3 dimana jika koneksi berhasil maka akan menampilkan *Inbox list* yang berisi *email – email* yang telah berhasil di *download*, kemudian user memilih *email* mana yang akan ditampilkan. Jika koneksi gagal maka akan ditampilkan peringatan. Dapat dilihat pada gambar 7.

Diagram *Activity* BalasEmail

Diagram *activity* Balas *Email* menunjukkan serangkaian proses aktivitas *user* melakukan proses membalas *email*. Dalam aktivitasnya terdapat lima *invocation* yang menunjukkan setiap proses ke proses berikutnya dan satu *decision* untuk proses berhasil tidaknya dalam melakukan koneksi terhadap POP3. Diagram ini memperlihatkan *user* melakukan koneksi POP3 dimana jika koneksi berhasil maka akan menampilkan *Inbox list* yang berisi *email – email* yang telah berhasil di *download*, kemudian user memilih *email* mana yang akan dibaca dan untuk dibalas. Jika koneksi gagal maka akan ditampilkan peringatan. Dapat dilihat pada gambar 8.

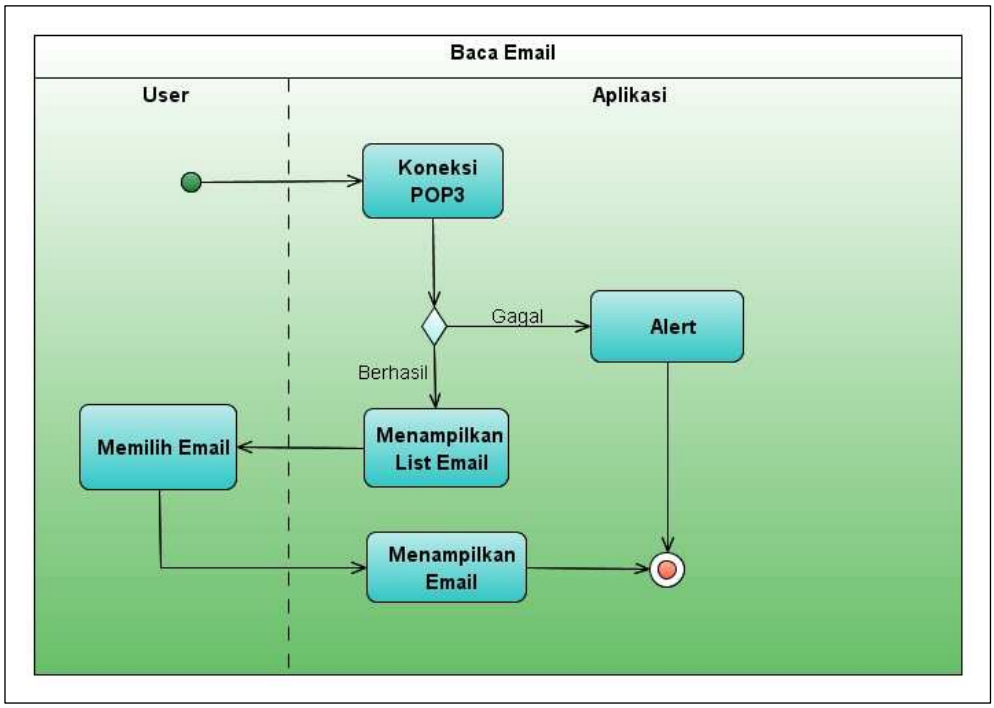
Diagram *Activity* TeruskanEmail

Diagram *activity* Teruskan *Email* menunjukkan serangkaian proses aktivitas *user* melakukan proses *forward email*. Dalam aktivitasnya terdapat enam *invocation* yang menunjukkan setiap proses ke proses berikutnya dan satu *decision* untuk proses berhasil tidaknya dalam melakukan koneksi terhadap POP3. Diagram ini memperlihatkan *user* melakukan koneksi POP3 dimana jika koneksi berhasil maka akan menampilkan *Inbox list*

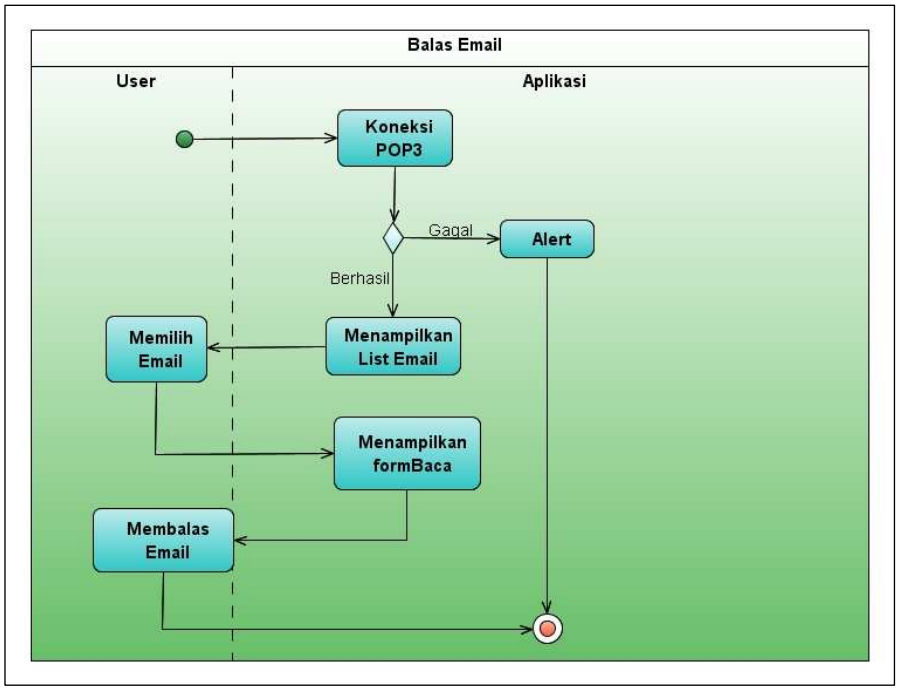
yang berisi *email-email* yang telah berhasil di *download*, kemudian user memilih *email* mana yang akan dibaca dan untuk diteruskan pada orang lain. Jika koneksi gagal maka akan ditampilkan peringatan (gambar 9).

Diagram *Activity* HapusEmail

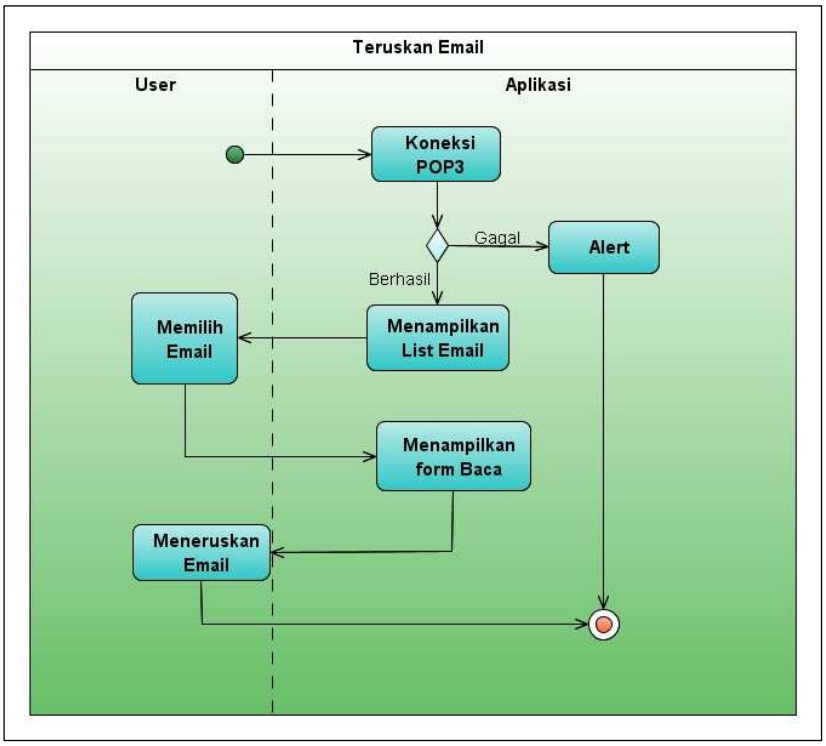
Diagram *activity* Hapus *Email* menunjukkan serangkaian proses aktivitas *user* melakukan proses hapus *email*. Dalam aktivitasnya terdapat enam *invocation* yang menunjukkan setiap proses ke proses berikutnya dan satu *decision* untuk proses berhasil tidaknya dalam melakukan koneksi terhadap POP3. Diagram ini memperlihatkan *user* melakukan koneksi POP3 dimana jika koneksi berhasil maka akan menampilkan *Inbox list* yang berisi *email – email* yang telah berhasil di *download*, kemudian *user* memilih *email* yang akan dihapus. Aplikasi akan mengambil parameter *email* yang akan dihapus dan aplikasi akan menghapus *email* yang telah dipilih . Jika koneksi gagal maka akan ditampilkan peringatan. Dapat dilihat pada gambar 10.



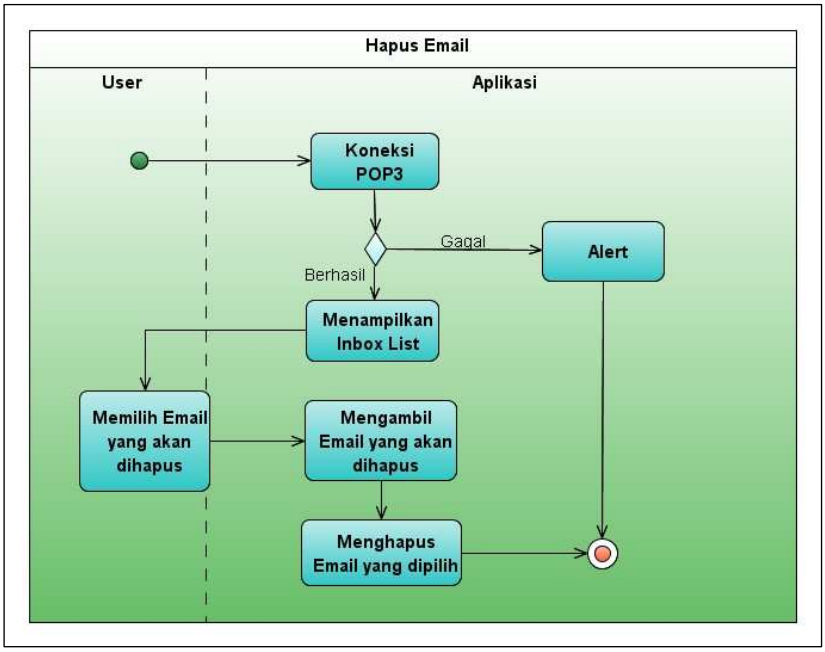
Gambar 7. Diagram *Activity* Baca *Email* Aplikasi *Email* menggunakan J2ME.



Gambar 8 Diagram *Activity* Balas *Email* Aplikasi *Email* menggunakan J2ME.



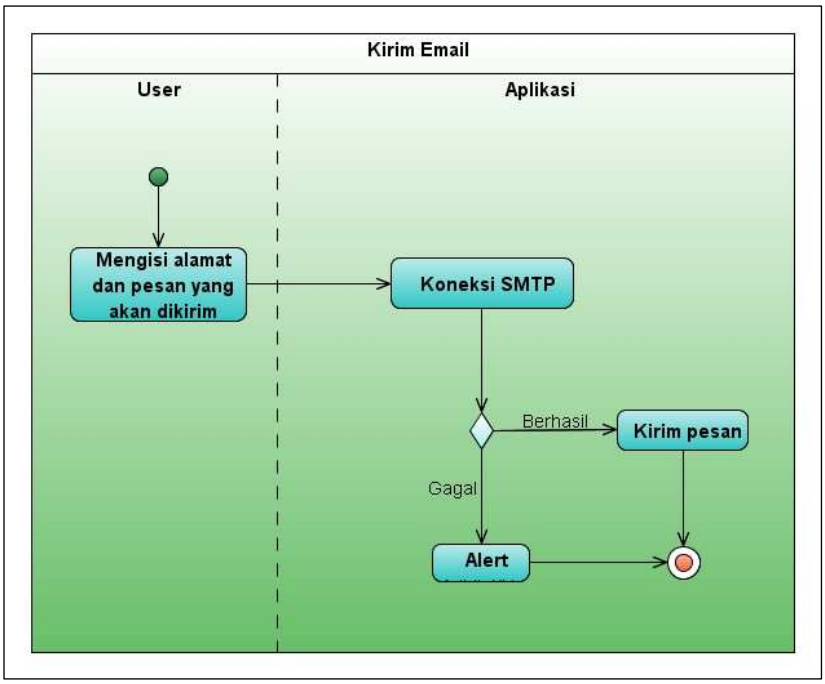
Gambar 9 Diagram *Activity* Teruskan *Email* Aplikasi *Email* menggunakan J2ME.



Gambar 10. Diagram *Activity* Hapus *Email* Aplikasi *Email* menggunakan J2ME.

Diagram *Activity* KirimEmail

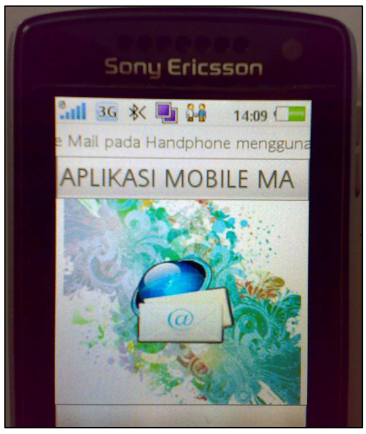
Diagram *activity* Kirim *Email* menunjukkan serangkaian proses aktivitas *user* melakukan proses pengiriman *email*. Dalam aktivitasnya terdapat empat *invocation* yang menunjukkan setiap proses ke proses berikutnya dan satu *decision* untuk proses berhasil tidaknya dalam melakukan koneksi terhadap SMTP. Diagram ini memperlihatkan *user* mengisi alamat dan pesan pada *email* yang akan dikirim. Aplikasi melakukan koneksi SMTP dimana jika koneksi berhasil maka pesan akan dikirim. Jika koneksi gagal maka akan ditampilkan peringatan.



Gambar 11. Diagram *Activity* Kirim *Email* Aplikasi *Email* menggunakan J2ME.

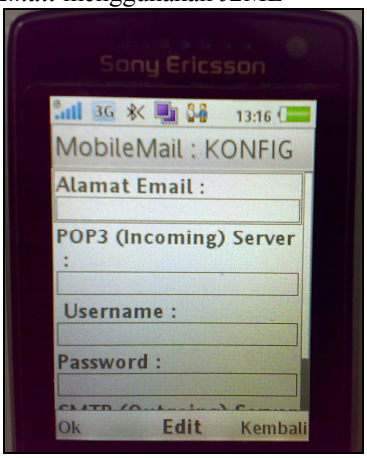
3 IMPLEMENTASI

Aplikasi berbasis *Email* menggunakan J2ME mempunyai sembilan *interface*, diantaranya :

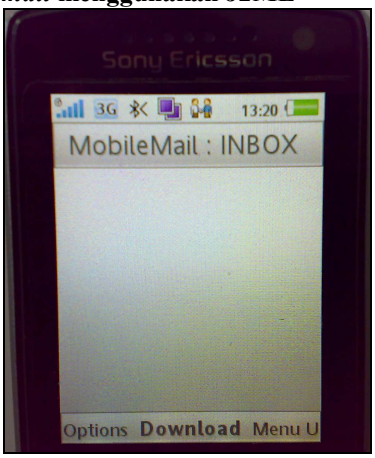


Gambar 12. Antarmuka ***Splash Screen*** Aplikasi *Email* menggunakan J2ME

Gambar 13. Antarmuka Menu Utama Aplikasi ***Email* menggunakan J2ME**



Gambar 14. Antarmuka Konfigurasi Aplikasi *Email* menggunakan J2ME



Gambar 15. Antarmuka *Inbox* Aplikasi *Email* menggunakan J2ME

a. **Splashscreen** merupakan *interface* yang muncul pertama kali pada saat membuka aplikasi (gambar 12)
b. **Antarmuka Menu Utama** merupakan *interface* yang mempunyai menu-menu untuk dihubungkan ke berbagai fungsi aplikasi (gambar 13).
c. **Antarmuka Konfigurasi** merupakan tampilan bagi *user* untuk melakukan penyetingan konfigurasi *email* (gambar 14)
d. **Antarmuka *Inbox*** merupakan tampilan bagi *user* untuk melakukan *download email* (gambar 15).
e. **Antarmuka Kirim *Email*** merupakan tampilan untuk *user* melakukan pengiriman *email* (gambar 16)

Gambar 16. Antarmuka **Kirim *Email*** Aplikasi *Email* menggunakan J2ME

4 KESIMPULAN

Kesimpulan pada skripsi ini adalah telah berhasil dibangun suatu Aplikasi *Email* menggunakan J2ME. Aplikasi ini dapat melakukan proses pengiriman dan penerimaan *email*. Dibuatnya Aplikasi *Email* menggunakan J2ME diharapkan dapat membantu *user* dalam melakukan proses pengiriman *email* maupun menerima *email* menggunakan *handphone*.

5 DAFTAR PUSTAKA

Fowler, Martin, 2005, *UML Distillled Edisi 3*, *Panduan Singkat Bahasa Pemodelan Objek Standar,* Andi Offset, Yogyakarta.

Hartono, J., 2002, *Pengenalan Komputer Dasar Ilmu Komputer, Pemrograman, Sistem Informasi dan Intelegensi Buatan*., Andy, Yogakarta.

Kadir, Abdul, 2002, *Pengenalan Sistem Informasi*, Andi Offset, Yogyakarta.

Kadir, Abdul, 2004, *Dasar Pemrograman Java 2*, Andi, Yogyakarta.

Kruchten, P., 2003, *The Rational Unified Process: An Introduction, Third Edition*, Addison Wesley.

Kruchten, P., dan Kroll, P., 2003, *Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, The, Addison Wesley.

Mardiono, T., 2006, *Membangun Solusi Mobile Business dengan Java*, Elex Media Komputindo, Jakarta.

Munawar, 2005, *Pemodelan Visual dengan UML*, Penerbit Graha Ilmu, Yogyakarta.

Prsetyo, D.D., 2004, *Mail Service Berbasis Java pada Server Windows dan Linux*, Elex Media Komputindo, Jakarta.

Purnomo, A.E., 2005, *POP3 Mail Retriever sederhana menggunakan Delphi*, Universitas Gajah Mada, Yogyakarta.

Shalahuddin, M., dan A.S., Rosa, 2006, *Pemrograman J2ME : Belajar cepat Pemrograman Perangakat Telekomunikasi Mobile*, Informatika, Bandung.

Suyoto, 2005, *Membuat Sendiri Aplikasi Ponsel*, Gava Media, Yogyakarta.

Thamura, Frans, http://ilmukomputer.com/2007/02/23/netbeans-open-source-java-ide-berbasiskan-swing/ (accessed 20 Agustus 2008)